C# : exceptions

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

C#

Exception

- C'est une erreur qui se produit pendant l'exécution de notre programme
- Une exception dans un programme implique généralement son arrêt d'exécution

C#

Comment faire pour poursuivre l'exécution ?

- Repérer les blocs pouvant générer une exception
- Capturer l'exception correspondante
- Afficher un message relatif à cette exception
- Continuer l'exécution

C#

Commençons par créer un nouveau projet dans `MaSolution`

- Dans l'`Explorateur de solutions`, faire clic droit sur `MaSolution`
- Aller à `Ajouter > Nouveau projet`
- Sélectionner `Application console`
- Cliquer sur `Suivant`
- Remplir les champs
  - `Nom` avec `CoursException`
  - `Solution` avec `MaSolution`
- Valider

`Exception` : exemple

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0;
            int y = 5 / x;
            Console.WriteLine(x);
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

`Exception` : exemple

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0;
            int y = 5 / x;
            Console.WriteLine(x);
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

Message affiché à l'exécution

Exception non gérée :
System.DivideByZeroException : 'Attempted to divide by zero.'.

`Exception` : exemple

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 0;
            int y = 5 / x;
            Console.WriteLine(x);
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

Message affiché à l'exécution

Exception non gérée :
System.DivideByZeroException : 'Attempted to divide by zero.'.

Constatation

- Le message `Fin de calcul` n'a pas été affiché
- La division par zéro déclenche une exception `DivideByZeroException`

C#

Comment faire pour capturer une exception ?

- Utiliser un bloc `try { ... } catch { ... }`
- Le `try { ... }` pour entourer une instruction susceptible de déclencher une exception
- Le `catch { ... }` pour capturer l'exception et afficher un message qui lui correspond

C#

`Exception` : exemple

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Exception : Division par zero ");
            }
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

C#

Message affiché à l'exécution

Exception : Division par zéro
Fin de calcul

C#

Message affiché à l'exécution

Exception : Division par zéro
Fin de calcul

Constatation

- L'exception a été capturée.
- Le message `Fin de calcul` a été affiché.

C#

Et si je ne connais pas le type d'exception

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception : Division par zero ");
            }
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

C#

Le même message sera affiché

Exception : Division par zéro
Fin de calcul

C#

Le même message sera affiché

Exception : Division par zéro
Fin de calcul

Explication

La classe `Exception` est la classe mère de toutes les classes d'exception.

C#

Utiliser des méthodes de la classe `Exception`

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Exception : " + e.Message);
            }
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

C#

Utiliser des méthodes de la classe `Exception`

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Exception : " + e.Message);
            }
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

Message affiché

Exception : Attempted to divide by zero.
Fin de calcul

Utiliser des méthodes de la classe `Exception`

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine(e.StackTrace); ;
            }
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

Utiliser des méthodes de la classe `Exception`

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine(e.StackTrace); ;
            }
            Console.WriteLine("Fin de calcul");
        }
    }
}
```

Message affiché

à CoursException.Program.Main(String[] args) dans C :/.../source/repos/MySolution/CoursException/Program.cs :ligne 12
Fin de calcul

C#

On a utilisé (ou vu) des exceptions prédéfinies

- `Exception`
- `DivideByZeroException`
- `IndexOutOfRangeException`

C#

On a utilisé (ou vu) des exceptions prédéfinies

- `Exception`
- `DivideByZeroException`
- `IndexOutOfRangeException`

On peut aussi définir nos exceptions personnalisées

C#

Commençons par créer une classe `Adresse` dans un dossier `Models`

```
namespace CoursException.Models
{
    internal class Adresse
    {
        public string Rue { get; set; }
        public string CodePostal { get; set; }
        public string Ville { get; set; }

        public Adresse(string rue, string ville, string codePostal)
        {
            Rue = rue;
            Ville = ville;
            CodePostal = codePostal;
        }
    }
}
```

C#

Supposons que

`codePostal` doit contenir exactement 5 chiffres

C#

Supposons que

`codePostal` doit contenir exactement 5 chiffres

Démarche à faire

- Créer notre propre exception (qui doit étendre la classe `Exception`).
- Dans le constructeur de `Adresse`, on lance une exception si `codePostal` ne contient pas 5 chiffres.

C#

Créons l'exception `IncorrectCodePostalException` **dans** `Exceptions`

```
namespace CoursException.Exceptions
{
    public class IncorrectCodePostalException : Exception
    {
        // le constructeur de cette nouvelle exception
        public IncorrectCodePostalException() :
                    base("Le code postal doit contenir exactement
                        5 chiffres")
        {
        }
    }
}
```

Modifions le constructeur de la classe `Adresse`

```
using CoursException.Exceptions;

namespace CoursException
{
    public class Adresse
    {
        public string Rue { get; set; }
        public string CodePostal { get; set; }
        public string Ville { get; set; }

        public Adresse(string rue, string ville, string codePostal)
        {
            if (codePostal.Length != 5)
            {
                throw new IncorrectCodePostalException();
            }
            CodePostal = codePostal;
            Rue = rue;
            Ville = ville;
        }
    }
}
```

Testons tout cela dans `Main()`

```
using System;
using CoursException.Models;
using CoursException.Exceptions;

namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Adresse a = new Adresse("paradis", "Marseille", "1300");
            }
            catch (IncorrectCodePostalException icpe)
            {
                Console.WriteLine(icpe.Message);
            }
        }
    }
}
```

Testons tout cela dans `Main()`

```
using System;
using CoursException.Models;
using CoursException.Exceptions;

namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Adresse a = new Adresse("paradis", "Marseille", "1300");
            }
            catch (IncorrectCodePostalException icpe)
            {
                Console.WriteLine(icpe.Message);
            }
        }
    }
}
```

Message affiché

Le code postal doit contenir exactement 5 chiffres.

C#

On peut rajouter une deuxième hypothèse

- `codePostal` doit contenir exactement 5 chiffres
- `rue` doit être une chaîne en majuscule

C#

Créons une deuxième exception `IncorrectStreetNameException` **dans** `Exceptions`

```
namespace CoursException.Exceptions
{
    public class IncorrectStreetNameException : Exception
    {
        public IncorrectStreetNameException() : base("Le nom de
            la rue doit être en majuscule")
        {
        }
    }
}
```

C#

Modifions le constructeur de la classe `Adresse`

```
namespace CoursException.Models
{
    public class Adresse
    {
        public string Rue { get; set; }
        public string CodePostal { get; set; }
        public string Ville { get; set; }

        public Adresse(string rue, string ville, string codePostal)
        {
            if (codePostal.Length != 5)
            {
                throw new IncorrectCodePostalException();
            }
            if (!rue.Equals(rue.ToUpper()))
            {
                throw new IncorrectStreetNameException();
            }
            CodePostal = codePostal;
            Rue = rue;
            Ville = ville;
        }
    }
}
```

Re-testons tout cela dans `Main()`

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Adresse a = new Adresse("Oddo", "Marseille", "13015");
            }
            catch (IncorrectCodePostalException icpe)
            {
                Console.WriteLine(icpe.Message);
            }
            catch (IncorrectStreetNameException isne)
            {
                Console.WriteLine(isne.Message);
            }
        }
    }
}
```

C#

On peut aussi fusionner les `catch`

```
namespace CoursException
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Adresse a = new Adresse("Oddo", "Marseille", "13015");
            }
            catch (Exception e) when
              (e is IncorrectCodePostalException ||
               e is IncorrectStreetNameException)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

C#

Hypothèse

Si on voulait afficher les valeurs qui ont déclenché l'exception dans le message

C#

Modifions la première exception `IncorrectCodePostalException`

```
namespace CoursException.Exceptions
{
    public class IncorrectCodePostalException : Exception
    {
        // le constructeur de cette nouvelle exception
        public IncorrectCodePostalException(string cp) :
                    base($"Le code postal { cp } doit contenir
                       exactement 5 chiffres")
        {
        }
    }
}
```

C#

Modifions la deuxième exception `IncorrectStreetNameException`

```
namespace CoursException.Exceptions
{
    public class IncorrectStreetNameException : Exception
    {
        public IncorrectStreetNameException(String rue) :
          base("Le nom de la rue '" + rue + "' doit être en
             majuscule")
        {
        }
    }
}
```

C#

Modifions le constructeur de la classe `Adresse`

```
namespace CoursException.Models
{
    public class Adresse
    {
        public string Rue { get; set; }
        public string CodePostal { get; set; }
        public string Ville { get; set; }

        public Adresse(string rue, string ville, string codePostal)
        {

            if (codePostal.Length != 5)
            {
                throw new IncorrectCodePostalException(codePostal);
            }
            if (!rue.Equals(rue.ToUpper()))
            {
                throw new IncorrectStreetNameException(rue);
            }
            CodePostal = codePostal;
            Rue = rue;
            Ville = ville;

        }
    }
}
```

C#

Pour tester

```
namespace CoursException
{
    class Program
    {
        static void Main(String[] args)
        {
            try
            {
                Adresse a = new Adresse("Oddo", "Marseille", "1301");
            }
            catch (Exception e) when
              (e is IncorrectCodePostalException ||
               e is IncorrectStreetNameException)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

C#

Pour tester

```
namespace CoursException
{
    class Program
    {
        static void Main(String[] args)
        {
            try
            {
                Adresse a = new Adresse("Oddo", "Marseille", "1301");
            }
            catch (Exception e) when
              (e is IncorrectCodePostalException ||
               e is IncorrectStreetNameException)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Message affiché

Le code postal ’1300’ doit contenir exactement 5 chiffres

C#

Exercice

Créer une nouvelle classe d'exception `AdresseException` pour fusionner et remplacer les deux exceptions `IncorrectCodePostalException` et `IncorrectStreetNameException`

C#

À utiliser lorsqu'on a une instruction à exécuter qu'une exception soit levée ou non

Exemple

```
namespace CoursException
{
    class Program
    {
        static void Main(String[] args)
        {
            int x = 5, y = 0;
            try
            {
                Console.WriteLine(x / y);
            }
            catch (Exception e)
            {
                Console.WriteLine("Division par zero");
            }
            finally
            {
                Console.WriteLine("Instruction exécutée systé
                   matiquement");
            }
        }
    }
}
```

C#

Remarque

Le bloc `finally` peut s'avérer intéressant si le `catch` contient un `return` qui forcera l'arrêt de l'exécution du code. Ce bloc (`finally`) sera toujours exécuté.